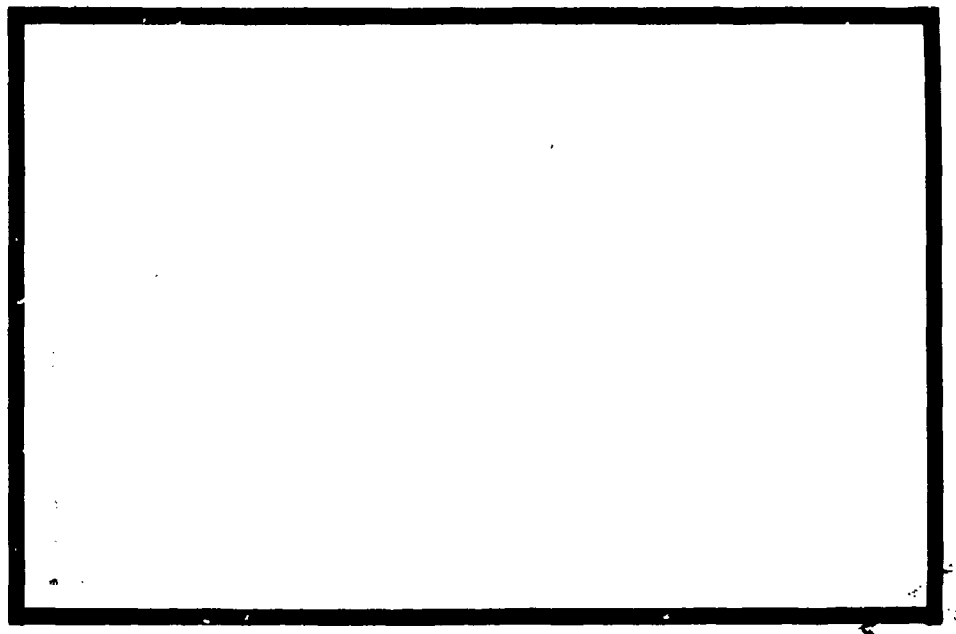
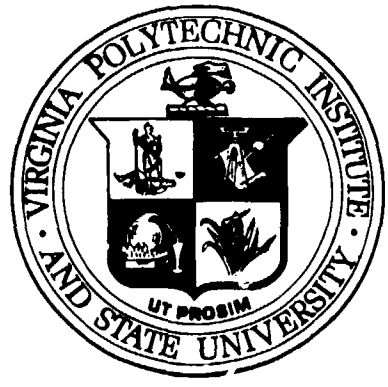
LEVEL II

(10)





Virginia Polytechnic Institute and State University

Computer Science
Industrial Engineering and Operations Research
BLACKSBURG, VIRGINIA 24061

CSIE-81-3

September 1981

10

SAM -- A CONFIGURABLE EXPERIMENTAL TEXT EDITOR FOR INVESTIGATING HUMAN FACTORS ISSUES IN TEXT PROCESSING AND UNDERSTANDING

Roger W. Ehrich

TECHNICAL REPORT

Prepared for
Engineering Psychology Programs, Office of Naval Research
ONR Contract Number N00014-81-K-0143
Work Unit Number NRSRO-101

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: CSIE-81-3	2. GOVT ACCESSION NO.: AD-A109 331
3. RECIPIENT'S CATALOG NUMBER:	
4. TITLE (and Subtitle): SAM -- A CONFIGURABLE EXPERIMENTAL TEXT EDITOR FOR INVESTIGATING HUMAN FACTORS ISSUES IN TEXT PROCESSING AND UNDERSTANDING	5. TYPE OF REPORT & PERIOD COVERED: Technical Report
	6. PERFORMING ORG. REPORT NUMBER:
7. AUTHOR(s): Roger W. Ehrich	8. CONTRACT OR GRANT NUMBER(s): N00014-81-K-0143
9. PERFORMING ORGANIZATION NAME AND ADDRESS: Computer Science, Virginia Polytechnic Institute & State University, Blacksburg, VA 24061	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: NRSRO-101
11. CONTROLLING OFFICE NAME AND ADDRESS: Office of Naval Research, Code 442, 800 North Quincy Street, Arlington, VA 22217	12. REPORT DATE: September 1981
	13. NUMBER OF PAGES: 30
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release; distribution unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

text, editor, commands, syntax, context, completion, human factors

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

There are numerous behavioral issues involving human-machine communications that are exemplified by text editing applications. Examples include scanning mechanisms and information location techniques, command language design, effect of language syntax on productivity and accuracy, and the relationship of syntax to the user task model. In general, there exist little data to support design goals, and in order to formulate studies of these issues a flexible text editing system is required. This report describes SAM, a reasonably sophisticated text editor that has table driven command language and

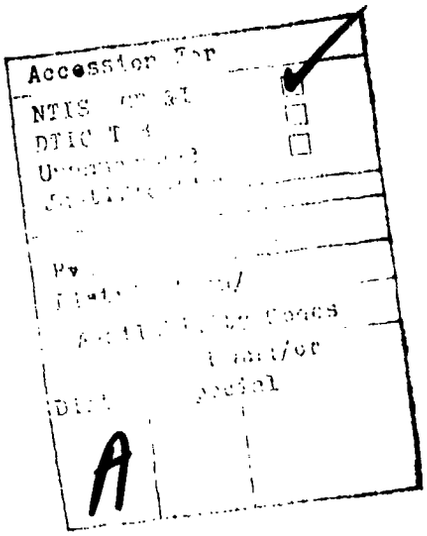
DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

S/N 0102-LF-014-6601

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

20. ABSTRACT

syntax that allows reasonably easy reconfiguration for special experiments. In particular, it is straightforward to insert the metering procedures required to measure specific aspects of human performance.



ACKNOWLEDGEMENTS

This research was supported in part by the Office of Naval Research under ONR Contract Number N00014-81-K-0143, and Work Unit Number NRSRO-101. The effort was supported by the Engineering Psychology Programs, Office of Naval Research, under the technical direction of Dr. John J. O'Hare.

TABLE OF CONTENTS

INTRODUCTION

In order to perform behavioral studies relating to text processing applications it is necessary to have an editor that can be configured quickly for specialized experiments. Since most editors are inherently complicated, an organization is needed that experimenters can modify for behavioral studies without extensive programming experience. The result was an editor called SAM, which is a context oriented line editor with a number of novel features and creature comforts that computer people are not used to seeing on an editor. Some of these ideas come from years of discussion, others come from existing editors, and yet others had their origins in an editing/human factors seminar held in the Computer Science Department at Virginia Tech in the spring of 1980. Thus, SAM has some novel features, both in its external behavior and in its internal structure.

Appearance

One of SAM's most novel features is a queue, which is the mechanism by means of which lines may be moved around from location to location within a file or between files. SAM allows a user to open two files, called the PRIMARY and SECONDARY files, and the SWITCH command allows the user to flip back and forth between the two files. The queue is the means of transferring lines from one file to another, and with the queue one can accomplish such tasks as reversing line order and reordering sections of text in arbitrary order.

SAM allows the user to edit syntactically incorrect commands, and most of the command operands are generalized to function upward or downward by supplying a signed operand. In the case of string operands, a single space or sign after the command name acts as the string delimeter. All commands, switches, and switch states can be specified by typing a unique prefix, and underscores may be used to separate multi-word commands.

SAM does not allow a user to type operands for commands that are incorrect. In fact, the user is warned aurally of erroneous input, and the number of bells heard is proportional to the severity of the error. If the command is correct but the operand is erroneous, the user is warned and is asked to complete the operand from the point where the erroneous input occurred. On the other hand, every effort is made to make sense of the user's input.

SAM is intended as a programming tool, although some may find its features amenable to certain text processing applications as well. The user is urged to make use of relative movements in locating information to be edited. Although it is easy to determine absolute line numbers and locate information by absolute addressing, the command structure is designed to minimize the use of absolute addressing.

Internals

SAM has 8300 lines of code, mostly FORTRAN 77, in 55 modules. Of these, the main storage system is manipulated with only 3 procedures - INSERT, DELETE, and NEXT. All messages are produced by one subroutine, and two subroutines - CPARSER and SPARSER, are responsible for all command and operand keywords. These are automatic completion algorithms that allow an experimenter to add, delete, and modify names simply by inserting them into alphabetized tables. Two subroutines - COMMAND and STORE - take care of the detailed handling of individual characters in edit and input mode. Finally, there are 11 modular parsers that plug into the code as needed to decode individual operands. These are table driven so that they may easily be modified by altering state transition tables. The rest of SAM consists of the code necessary to communicate with the user's terminal open and close files, and perform the actual editing functions.

In order to specialize SAM for an experiment, the experiment designer

1) Inserts the command names into an alphabetized list in CPARSER.
2) Selects the appropriate operand parser for each command from those provided (or modifies some of the existing parsers).
3) Selects the semantic routine for each command that actually performs the editing function. Functions not provided by SAM must be added, and this requires on the average about 50 lines of code per new command.
4) Decides what should be metered and inserts calls to the metering procedure in appropriate places.

The time required to modify SAM varies from a few hours for simple command renaming or elimination to a week or two if new commands are to be designed.

Input mode

CR The standard line terminator. When the cursor is at the left margin, it causes a mode change to edit mode.

LF Line terminator ($J on terminals without LF).

ESC Escape character preceding any character except those trapped by VMS will cause that character to be entered as input.

Command mode

There are two classes of commands - immediate commands and standard commands. Immediate commands are single character commands that are executed immediately when a key is struck at the beginning of a new line. There are 6 immediate commands:

. Type the the current line.

Type the number of the current line.

+ Move the line pointer to the next line and echo this line.

- Move the line pointer to the previous line and echo this line.

? Type the default help for SAM

CR Switch between edit and input modes and give a bell signal.

There are 31 standard commands, and several of these have synonyms. In the following table, capital letters denote minimal prefixes, and underscores may or may not be used as punctuation, as desired.

=, REPEat
ANchor
A, ARea
APpend
C, CHange
CQ, CLear_queue
COmbine
D, DELETE
DS, DELETE_To_string
EL, EDit_line

```
File
GS, GEt_secondary
HELp
HEX
Locate
Next
PD, POP_And_delete
POP
Q, QUEUE
QD, QUEUE_And_delete
QUIt
RETurn
R, REPLace
S, SWitch
SAve
SEt
SPlit
STatus
T, TYPE
TQ, TYPE_Queue
nnn (Set pointer to line nnn)
```

Command Completion

Whenever a unique command prefix is detected, the command is completed automatically. The user need not be aware of this, since if the rest of the command is typed after completion, the extra characters are ignored.

Character Set

Six characters are trapped by the VAX terminal driver, and these characters have the normal VAX functions. These are

```
CNTRL  C   ETX   (Returns control to the command processor)
CNTRL  O   SI    (Cancels the current output operation)
CNTRL  Q   DC1   (XON=Resume output)
CNTRL  S   DC3   (XOFF=Suspend output)
CNTRL  X   CAN   (Cancels any read and purges type-ahead)
CNTRL  Y   EM    (Returns control to the command processor)
```

In addition, the editor uses the following characters.

```
           CR    (Line terminator)
CNTRL  U   NAK   (Line delete)
           DEL   (Character delete)
           BS    (Character delete)
           ESC   (Escape)
```

		TAB	(Horizontal tabs)
CNTRL	J	LF	(Line terminator)

Any of these last 7 characters can be inserted into a line by preceding the character by ESC. In addition to the above characters, * is used to represent numerical infinity wherever the context makes sense.

Tabs

SAM has two types of tab stops, setable with the SET command. Pseudo tabs cause echoing of SP characters and insertion of SP characters into the text file. This feature facilitates indentation of programs. True tabs cause echoing of SP characters and insertion of HT into the text file.

Command pre-parsing

In edit mode, SAM checks for command validity as soon as a command is typed. If a command is invalid, two bells are given, and further typing is blocked. The user may backspace to make corrections or cancel the line. Valid commands:

1) May be preceded by non-printing characters
2) Must not contain spaces
3) May contain underscores for clarity
4) Must be a complete command or an unambiguous prefix
5) Must be followed by a non-digit if the command is a number
6) Must be followed by a non-letter if the command is a character string

Startup Command File

SAM has the capability for executing a startup command file when an editing session begins. This command file is executed just before SAM types EDIT or INPUT after the editor is invoked. When SAM is invoked, it searches the logical name tables in VMS

for the name, START_SAM. If this name is present, its translation is interpreted to be the name of the command file to be executed. As an example, suppose that init.sam was the command file to be executed. Then, the logical name assignment is made by typing the VMS command

ASSIGN init.sam START_SAM

Equivalently, this VMS command can be placed in the user's login.com.

All SAM commands can be executed by typing them in a file, one command per line. Any command line with improper syntax will be skipped.

=, REPEat

Repeats the most recent successful command. Immediate commands cannot be repeated by using the repeat command.

A, ARea { N | * }

Displays the context of ±N lines around the current line. If N is omitted, a default value dependent upon the terminal data rate is selected. The current line is unchanged.

ANchor

The location of the current line is saved by the editor. No matter what changes are made in the file, the RETURN command will return the editor to the anchored line unless it has been deleted.

APpend { STRING }

If no string is given, places the cursor at the end of the current line and enters temporary input mode. If a string is specified it is appended to the current line.

C, CHange { + | - | N | +N | -N | * | +* | -* } [D S1 [D S2 [D]]]

On N lines, including the current line, replace string S1 with S2. The delimeter, D, may be any printing character that does not appear in either S1 or S2. The changes are made for the first, last, or all occurrences of S1, depending upon the setting of the change switch. The current line remains the same. The default is to change 1 line.

CQ, CLear_queue { N | * }

Clears the queue of the specified number of lines. If no N is specified, the entire queue is cleared.

COmbine

Appends the next line to the current line and replaces both by the single new line. The current line becomes the new line.

D, DELETE { + | - | N | +N | -N | * | +* | -* }

Deletes a specified set of lines including the current line. The current line becomes the next undeleted line in the direction of the deletions. The default is 1.

DS, DELETE_To_string [STRING | +STRING | -STRING]

Deletes lines, including the current line up to but not including the first line containing STRING. The command has no effect if no line containing STRING is found. Otherwise the current line becomes the one that contains STRING. The only characters that may precede STRING are a single SP, a sign, or a SP and a sign.

EL, EDit_line

Enters a special editing mode for the current line. Space to the point where a change is desired. Then type:

```
  C - Change current character and advance
 CR - Terminate edit line mode
  D - Delete current character and advance
  I - Insert string before next character
  R - Reverse the case of current character and advance
 ¢U - Re-edit the current line
```

File { + | FILENAME | +FILENAME }

Files the primary file and terminates the editor. If no FILENAME is specified the current file name is used. A new version number is generated if the + symbol is specified.

GS, GEt_secondary [FILENAME]

Opens the named secondary file for editing. Any previously opened secondary file is eliminated from SAM's internal storage without writing it back out to permanent storage.

The secondary file can be stored permanently only by using the SAVE command.

HELp [COMMAND]

Types out a command list or provides help on a command if one is specified. If help is requested after making an error in a command operand, help will default to the help for that command. Help on SET command switches is available by typing, HELP SET <switch>.

HEX [MNEMONIC | hh | ddd]

If no operand is specified, the ASCII mnemonics for the characters in the current line are given. If the operand is an ASCII mnemonic, a 2 digit hex number, or a 3 digit decimal number, all 3 representations are returned. If the character is a control character, the terminal control code is also returned.

Locate [STRING | +STRING | -STRING]

Locates the first line containing the specified string. There is no effect if the string is not found. The only characters that may precede STRING are a single SP, a sign, or a SP and a sign. The number of lines traversed in locating the desired line is printed on the terminal.

Next [+ | - | N | +N | -N | * | +* | -*]

Moves the current line pointer N lines in the specified direction. The default is 1.

PD, POP_And_delete { + | - | N | +N | -N | * | +* | -* }

Pops the specified number of lines from the queue into the text after the current line and deletes the popped lines from the queue. The current line is the last one popped from the queue. The default is 1. Upward pops will result in reversed line order.

POP { + | - | N | +N | -N | * | +* | -* }

Pops the specified number of lines from the queue into the text after the current line. The current line is the last one popped from the queue. The default is 1. Upward pops will result in reversed line order.

Q, QUEUE { + | - | N | +N | -N | * | +* | -* }

Places the specified number of lines, starting with the current line, onto the queue. The current line becomes the line following the last one placed on the queue. The default is 1.

QD, QUEUE_And_delete { + | - | N | +N | -N | * | +* | -* }

Places the specified number of lines, starting with the current line, onto the queue and deletes them from the text. The current line becomes the one after the last one deleted in the direction of the deletion. The default is 1.

QUIt

Leave the editor without writing files to permanent storage. If changes have been made, the user is asked to verify the request.

RETurn

Makes the editor return to the anchored line. The number of lines traversed in returning to the anchored line is printed on the terminal.

R, REPLace { + | - | N | +N | -N | * | +* | -* }

Deletes the specified lines, starting with the current line and enters input mode. If the deletion was an upward deletion, the inserted lines go above the current line. The current line becomes the last inserted line. The default is 1.

S, SWitch

The editing environment is toggled back and forth between the PRIMARY and SECONDARY files.

SAve { + | FILENAME | +FILENAME }

Files the current working file (PRIMARY or SECONDARY). If no FILENAME is specified the current file name is used. A new version number is generated if the + symbol is specified. The current file name will become the name of the file just written.

SEt [SWITCH { STATE } | SWITCH { LIST }]

Allows the user to alter the internal behavior of the editor. A LIST is a sequence of numbers separated by SPACEs. If a number or state is omitted, the editor default value is used. The SWITCHes and their STATEs are

ANsi_tabs Sets tabs in every 8 columns.
ARea { nnn } Sets the default context for the ARea command.
Bells { on | off } Sets audible notification of syntactic errors.
CHange { first | last | all } Affects the CHange command.
CLear_tabs { LIST } Clears tabs at specified locations.
COnvert_tabs { on | off } Sets echo of true tabs to SP characters.
Default_tabs { nnn } Tab in column 9 and pseudo tabs every nnn.
First_column { nnn } Sets the first column for editing commands.
Hard_copy { on | off } Sets echoing for hard copy terminals.
LAst_column { nnn } Sets the last column for editing commands.
LIne_length { nnn } Sets the maximum line length for files.
PArenthesis_count { on | off } Sets paren counting in input mode.
PSeudo_tabs { LIST } Sets pseudo tabs at specified positions.
Screen_width { nnn } Sets the terminal line length.
TAbs { LIST } Sets true tab stops at specified positions.
TEk { on | off } Sets screen handling for Tektronix terminals.
Upper_case { on | off } Sets conversion to upper case.

SPlit { STRING }

Splits the current line into two lines using the cursor if STRING is not specified or before STRING if STRING is specified. The current line will be the second part of the split line. The only character that can precede STRING is a single SP.

STatus

Displays current file names, identifies the working file, prints the current file lengths, and displays the state of the editing switches.

T, TYPE { N | * }

Types on the screen the specified lines, starting with the current line. The current line is the last one typed. The default is 1.

TQ, TYPE_Queue { N | * }

Types on the screen the specified lines from the queue. The default is 1.

N

Sets the line pointer to the line whose current line number is N. 0 is the empty line above the file, and * is the empty line below the file, since * represents numerical infinity.

SAM's STORAGE SYSTEM

Although numerous procedures support SAM's storage system, only three are used to enter or retrieve information. These are

```
INSERT (direction)
NEXT   (status,increment)
DELETE (status,increment)
```

where status is a byte that determines when the line pointer is at the top or bottom of the file and the other variables are of type *2.

INSERT requires that the line to be inserted is in the global array, input_buffer, and that icnt, the line length, be set to the line length. NEXT and DELETE return the current line to the input_buffer. The arguments can be positive or negative, and it is not possible to alter or delete the two special lines, TOF: and BOF: that are always present in the file.

The storage system itself is a virtual storage system that consists of three data structures -- the page tables, working storage for both primary and secondary files, and a queue. Upon entering SAM a delay is observed as the files are opened and as the source files are moved into the storage system. Once in the storage system, files can be edited very quickly. Source files are closed as soon as the storage system has been loaded, and the only open files are SAM internal files. If the system crashes or if the user exits with a ¢Y, the internal files remain intact and current up to the last time the editor paged. Upon reentry, SAM

notices the presence of these undeleted internal files and asks the user if he wishes to recover them.

SAM requires a minimum of 18 pages of disk storage and at least 36 pages if both primary and secondary files are to be used.

The Page Table

The page table is a physically sequential list of the numbers of the pages of working storage. Each entry is a *2 page number, and a 0 byte separates currently allocated pages from free pages. A number of procedures are provided for entering and accessing information in the page table. These are:

```
READ_PT     (n)
WRITE_PT    (n)
INIT_PT     (n)
QUERY_PT    (status,code,n,logical,physical,length)
ENTER_PT    (n,length)
SIZE_PT     (n,begin,current,end,total)
RESTORE_PT  (n)
EXPAND_PT   (status,n,physical)
SHRINK_PT   (n,pages)
READ_WORK   (n,page)
WRITE_WORK  (n,page)
```

Page Contents

Each page of working storage consists of 109 24-byte fragments. Each text line occupies several fragments, depending upon its length. Each fragment has a *2 length field and both forward and backward pointers. The length field of the first fragment of a line contains the line length, and other fragments of a line have a length field containing -1. The backward

pointer of the first fragment of the first line of a page and the forward pointer of the last fragment of the last line of a page contain 0. No page in the storage system may be entirely empty. Also stored with each page are the following *2 variables:

```
fline  - Pointer to the first fragment of the first line
ffree  - Pointer to the free space list
nlines - The number of lines in the page
nfree  - The number of free fragments
lline  - Pointer to the first fragment of the last line
```

In addition to the variables above associated with a page, there are other global variables that contain important information. These include:

```
cfile - The number of the currently active file
cfrag - Pointer to the first fragment of the current line
cptr  - The number of the current line in the page
cline - The number of the current line in the file
```

Paging

When an insertion is attempted in a page that is full, a procedure called SPLIT removes several lines from the current page, including the current line, and places them in a new page. The new page will contain between 10% and 50% of the lines from the old page. The advantage of the page table is that a line can be retrieved quickly from any location in the file and that deletions are also very fast, no matter how many lines are deleted. In the first implementation there is no provision for merging pages that have low occupancy. Under the assumptions that lines occupy an average of two fragments and assuming 75% page occupancy, the storage system will hold 32,767 lines, which is the address space of the editor.

The page tables for both the primary and secondary files consist of single 1534 byte records, and working storage has up to 766 pages of 3071 bytes for each file. Working storage makes use of direct access so that pages may be located without reading through the entire file.

File Recovery

SAM's internal working files are deleted upon normal exit from the editor. Should the system crash or should the user inadvertently leave SAM by typing ¢Y, the internal files are not deleted. Upon reentry into SAM, if these files are present, SAM will ask whether the user wishes to continue editing them. The user should respond with an immediate subcommand, y or n. The working files are updated only when the editor pages. Normally the SAVE command would be used if the user wishes to guarantee the currency of the work, though SAM can be forced to page by typing the SWITCH command.

I/O Services

All of SAM's i/o services are provided through assembly language subprograms that call the VAX Record Management System. The lowest level procedures are:

```
EXPAND_FNAME
OPENER
CLOSER
READ_SRC
WRITE_SRC
PAGEIO
STACK
```

SAM ignores all fixed control fields of text input files, but it writes sequentially numbered fixed control fields into its output files. The fixed control accounts for the sequential statement numbers that appear when a SAM-edited file is printed.

Parsers

SAM has 13 modular parsers, 2 of which are table-driven keyword analyzers and the remainder of which are finite state parsers. The keyword analyzers are

C_PARSER
S_PARSER

and the operand parsers are

CHANGE
EMPTY
FILE
HEX
NUMBER
POS_NUMBER
SIGNED_NUMBER
SIGNED_FILE
STRING
SIGNED_STRING
SWITCH

The operand parsers are adequate for parsing all command operands, and if new commands are created, these parsers can be used. C_PARSER and S_PARSER are used to decode command names and switches, respectively, on the basis of unique prefix. New command names or switches can be added simply by including them in alphabetized tables.

Syntax

One of the most difficult aspects of the design of SAM's syntax was determining a rule for delimiting a string. Except in the CHANGE command where string delimiters are printing

characters, a string operand is delimited from preceding characters by a single space. Also, if it makes sense, other implied delimiters can be used. For example, L90 makes sense because 9 is not part of any command containing characters; therefore 9 must be the first character of the operand string for the LOCATE command. Lab, however, appears to the syntax analyzer as an invalid command.

SAM's SCREEN HANDLER

SAM's character handling is one of its most important features, and SAM relies heavily upon full duplex communications. SAM interprets characters one by one as they are typed, and the response to each character is computed individually. While SAM has been implemented to be terminal independent, three broad classes of terminals - scrolling CRT's, non-scrolling but paged terminals, and hardcopy terminals - are supported. Two procedures do most of SAM's character interpretation:

STORE
COMMAND

STORE does character handling for input mode, and COMMAND does character handling in edit mode. These procedures are supported by three bottom level i/o procedures

PUT_CHARS
XMITCHARS
GET_CHARS

and the following service procedures

ECHO
SCREEN
TYPE_LINE
MSG
NEXT_TAB
TYPE_NUMBER
UPPER
QUIT
CLASS

In input mode, each character is interpreted, echoed, and stored as necessary in the input buffer by STORE. In addition, STORE keeps track of the physical location of each character on the screen so that it knows how to backspace over tabs and so that if

physical screen wraparound occurs, the screen line count can be updated. When SAM is called it sends an escape sequence to test for a Tektronix 4012/13 display. If the display is a Tektronix, SAM remembers the fact and does special screen handling, since the Tektronix will not scroll. If hardcopy or Tektronix mode is set, prompting and backspace handling change since no selective erase is possible on such terminals. Also, in Tektronix mode, each time the screen is cleared, a dither in the range 0 to 19 is subtracted randomly from the cursor's vertical home position to reduce screen degeneration.

In edit mode, character handling changes slightly. Every nonprinting character in the command input buffer is echoed as a SP character so that the user can tell exactly how many characters are in the buffer. As characters are typed, C_PARSER attempts to locate the end of the command. When it does, if the command is ambiguous or in error, the user is notified and, in the latter case, prevented from further typing. If the command is correct, no further parsing occurs until the line is complete. If the operand parser finds an error, the command line is rewritten up to the point where the error was detected so that the user can correct the command.

All user messages are transmitted to the terminal by the MSG subroutine. Thus, it is straightforward to change the communication from SAM to the user.

Screen clearing and special functions are provided by the SCREEN subroutine which, for the Tektronix, checks the line count each time it is called to determine whether there is space on the screen for the next line. Since ECHO is the only procedure in SAM to echo linefeed characters, ECHO is chiefly responsible for counting lines. STORE, COMMAND, and TYPE_LINE also increment the line count if physical wraparound occurs, since they keep track of screen location. TYPE_LINE has principal responsibility for calling SCREEN to test for available screen space. SCREEN must be called directly at all other points in SAM where output is sent to the screen with other output services.

442:MAT:716:maf
81u442-390

OFFICE OF NAVAL RESEARCH

Code 442

TECHNICAL REPORTS DISTRIBUTION LIST

OSD

CDR Paul R. Chatelier
Office of the Deputy Under Secretary
of Defense
OUSDRE (E&LS)
Pentagon, Room 3D129
Washington, D.C. 20301

Department of the Navy

Leader
Engineering Psychology Programs
Code 442
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217 (5 cys)

Leader
Communication & Computer Technology
Code 240
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Leader
Manpower, Personnel and Training
Code 270
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Dr. A. Meyrowitz
Information Systems Program
Code 411-IS
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Department of the Navy

Special Assistant for Marine
Corps Matters
Code 100M
Office of Naval Research
800 North Quincy Street
Arlington, VA 22217

Department of the Navy

Commanding Officer
ONR Eastern/Central Regional Office
ATTN: Dr. J. Lester
Building 114, Section D
666 Summer Street
Boston, MA 02210

Commanding Officer
ONR Branch Office
ATTN: Dr. C. Davis
1030 East Green Street
Pasadena, CA 91106

Commanding Officer
ONR Western Regional Office
ATTN: Dr. E. Gloye
1030 East Green Street
Pasadena, CA 91106

Office of Naval Research
Scientific Liaison Group
American Embassy, Room A-407
APO San Francisco, CA 96503

Director
Naval Research Laboratory
Technical Information Division
Code 2627
Washington, D.C. 20375 (6 cys)

Dr. L. Chmura
Code 7503
Naval Research Laboratory
Washington, D.C. 20375

Dr. Michael Melich
Communications Sciences Division
Code 7500
Naval Research Laboratory
Washington, D.C. 20375

Dr. Robert G. Smith
Office of the Chief of Naval
Operations, OP987H
Personnel Logistics Plans
Washington, D.C. 20350

Department of the Navy

Dr. Jerry C. Lamb
Combat Control Systems
Naval Underwater Systems Center
Newport, RI 02840

Naval Training Equipment Center
ATTN: Technical Library
Orlando, FL 32813

Human Factors Department
Code N215
Naval Training Equipment Center
Orlando, FL 32813

Dr. Alfred F. Smode
Training Analysis and Evaluation
Group
Naval Training Equipment Center
Code N-00T
Orlando, FL 32813

Dr. R. Neetz
Code 1226
Naval Missile Test Center
Pt. Mugu, CA 93042

Dr. Albert Colella
Combat Control Systems
Naval Underwater Systems Center
Newport, RI 02840

Dr. Gary Poock
Operations Research Department
Naval Postgraduate School
Monterey, CA 93940

Dean of Research Administration
Naval Postgraduate School
Monterey, CA 93940

Dr. A. L. Slafkosky
Scientific Advisor
Commandant of the Marine Corps.
Code RD-1
Washington, D.C. 20380

Dr. Thomas McAndrew
Code 32
NUSC-New London
New London, CT 06320

Department of the Navy

HQS, U.S. Marine Corps.
ATTN: CCA40 (MAJOR Pennell)
Washington, D.C. 20380

Commanding Officer
MCTSSA
Marine Corps. Base
Camp Pendleton, CA 92055

Chief, C^3 Division
Development Center
MCDEC
Quantico, VA 22134

Commander
Naval Air Systems Command
Human Factors Programs
NAVAIR 340F
Washington, D.C. 20361

Commander
Naval Air Systems Command
Crew Station Design,
NAVAIR 5313
Washington, D.C. 20361

Commander
Naval Electronics Systems Command
Human Factors Engineering Branch
Code 4701
Washington, D.C. 20360

CAPT Darrell D. Dempster, SC, USN (Ret)
System Management American Corporation
1745 Jefferson Davis Highway
Arlington, VA 22202

Dr. Mel C. Moy
Code 302
NPRDC
San Diego, CA 92152

Mr. Ramon L. Hershman
Code 302
NPRDC
San Diego, CA 92152

Navy Personnel Research and
Development Center
Planning & Appraisal
Code 04
San Diego, CA 92152

442:MAT:716:maf
81u442-390

Department of the Navy

Navy Personnel Research and
Development Center
Management Systems, Code 303
San Diego, CA 92152

Navy Personnel Research and
Development Center
Performance Measurement &
Enhancement
Code 309
San Diego, CA 92152

LCDR Stephen D. Harris
Code 6021
Naval Air Development Center
Warminster, PA 18974

Dr. Julie Hopson
Human Factors Engineering Division
Naval Air Development Center
Warminster, PA 18974

Dean of the Academic Departments
U.S. Naval Academy
Annapolis, MD 21402

Mr. John Impagliazzo
Code 101
NUSC - Newport
Newport, RI 02840

Walter P. Warner
Code K02
Strategic Systems Dept.
Naval Surface Weapons Center
Dahlgren, VA 22448

Dr. Thomas Fitzgerald
Code 101
NUSC - Newport
Newport, RI 02840

Department of the Air Force

Chief, Systems Engineering Branch
Human Engineering Division
USAF AMRL/HES
Wright-Patterson AFB, OH 45433

Department of the Air Force

Air University Library
Maxwell Air Force Base, AL 36112

Foreign Addressees

North East London Polytechnic
The Charles Myers Library
Livingstone Road
Stratford
London E15 2LJ
ENGLAND

Dr. Kenneth Gardner
Applied Psychology Unit
Admiralty Marine Technology
Establishment
Teddington, Middlesex TW11 OLN
ENGLAND

Director, Human Factors Wing
Defence & Civil Institute of
Environmental Medicine
Post Office Box 2000
Downsview, Ontario M3M 3B9
CANADA

Dr. A. D. Baddeley
Director, Applied Psychology Unit
Medical Research Council
15 Chaucer Road
Cambridge, CB2 2EF
ENGLAND

Professor B. Shackel
Department of Human Sciences
University of Technology
Loughborough, LEICS. LE11 3TU
ENGLAND

Other Government Agencies

Defense Technical Information Center
Cameron Station, Bldg. 5
Alexandria, VA 22314 (12 cys)

Dr. Craig Fields
Director, Cybernetics Technology
Office
Defense Advanced Research Projects
Agency
1400 Wilson Blvd.
Arlington, VA 22209

442:MAT:716:maf
81u442-390

Other Government Agencies

Dr. M. Montemerlo
Human Factors & Simulation
Technology, RTE-6
NASA HQS
Washington, D.C. 20546

Other Organizations

Dr. Jesse Orlansky
Institute for Defense Analyses
400 Army-Navy Drive
Arlington, VA 22202

Dr. T. B. Sheridan
Department of Mechanical Engineering
Massachusetts Institute of Technology
Cambridge, MA 02139

Dr. Arthur I. Siegel
Applied Psychologocal Services, Inc.
404 East Lancaster Street
Wayne, PA 19087

Dr. Robert T. Hennessy
NAS - National Research Council
JH #819
2101 Constitution Ave., N.W.
Washington, D.C. 20418

Mr. Edward M. Connelly
Performance Measurement
Associates, Inc.
410 Pine Street, S.E.
Suite 300
Vienna, VA 22180